UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de un chatbot para modelado colaborativo en Slack

Pablo Rísquez Almodóvar
Tutor: Sara Pérez Soler
Ponente: Esther Guerra Sánchez

JULIO 2019

Desarrollo de un chatbot para modelado colaborativo en Slack

AUTOR: Pablo Rísquez Almodóvar
TUTOR: Sara Pérez Soler
PONENTE: Esther Guerra Sánchez

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
JULIO de 2019

Resumen

Este Trabajo Fin de Grado habla sobre la creación de un Bot que integrará la API de modelado SOCIO con la aplicación empresarial SLACK.

La interacción entre el bot y la API de modelado se gestiona a través de peticiones por las cuales podremos gestionar, modificar y visualizar todos los cambios que puedan haberse dado en un proyecto pudiendo obtener detalles sobre el proyecto como pueden ser usuarios del proyecto que pueden leer o escribir en el así como obtener imágenes en tiempo real de alguna posible modificación por parte de otros usuarios o incluso realizar consensos y votaciones entre diferentes ramas para la elección del óptimo.

Este proyecto incluye la interacción con diferentes sistemas, redes y usuarios para un correcto y óptimo funcionamiento con la API, además de la labor de creación e integración de un bote que nos ayudara de una manera intuitiva, fácil y visual sobre la gestión de un proyecto, su evolución y los diferentes sucesos que pueden ocurrir en el mismo. Todo esto a través de una aplicación empresarial que está tomando mucha fuerza en el mercado, SLACK, la cual podría resumirse como una aplicación IRC donde habrá diferentes canales y los usuarios podrán comunicarse unos con otros. La gran ventaja de SLACK es la integración de diferentes aplicaciones y bits que aumentan la productividad de la misma pudiendo automatizar tareas y Mostar avisos a los usuarios que lo deseen.

En primera instancia se explicará que es un Bot y que papel puede desempeñar en las labores de interacción de un usuario y un sistema. Además, se mostrarán que tipos de sistemas y cómo actúan para un correcto funcionamiento de un sistema global que integra varios subsistemas y componentes digitales.

Se analizará y se mostrará todo el proceso técnico que se ha tenido que desempeñar para llevar a cabo la realización del Bot, así como todas las conexiones de los sistemas que se encuentran en la interacción del usuario con el Bot.

Summary

This Final Degree Project talks about the creation of a Bot that will integrate the SOCIO modeling API with the SLACK business application.

The interaction between the bot and the modeling API is managed through requests by which we can manage, modify and visualize all the changes that may have occurred in a project, being able to obtain details about the project, such as users of the project who can read or write in it as well as obtain images in real time of any possible modification by other users or even make consensuses and votes between different branches for the election of the optimum.

This project includes the interaction with different systems, networks and users for a correct and optimal functioning with the API, in addition to the task of creating and integrating a boat that will help us in an intuitive, easy and visual way about the management of a project, its evolution and the different events that may occur in it. All this through a business application that is taking a lot of force in the market, SLACK, which could be summarized as an IRC application where there will be different channels and users will be able to communicate with each other. The great advantage of SLACK is the integration of different applications and bits that increase the productivity of it, being able to automate tasks and display alerts to users who wish to do so.

In the first instance it will be explained that it is a Bot and what role it can play in the work of interaction of a user and a system. In addition, they will show what types of systems and how they work for a correct functioning of a global system that integrates several subsystems and digital components.

It will analyze and show all the technical process that has had to perform to carry out the realization of the Bot, as well as all the connections of the systems that are in the user's interaction with the Bot.

Palabras clave

API

SOCIO

VPS

Slack

Modelado

Agradecimientos

A mi familia, por su apoyo y ayuda en las situaciones difíciles. A mis compañeros, porque sin ellos el camino habría sido diferente, a mis amigos porque nunca dejaron de creer en mí y al profesorado de la universidad, por sus grandes lecciones.

INDICE DE CONTENIDOS

1 Introducción 3
1.1 Motivación 3
1.2 Objetivos 4
1.3 Organización de la Memoria 4
2 Estado del arte 5
2.1 Chatbots 5
2.2 Aplicaciones de modelado y aplicaciones colaborativas 7
3 Componentes del sistema 9
3.1 API MODELADO 9
3.2 SLACK 11
3.3 VPS 12
3.4 Botones interactivos 13
3.5 Chatbots 13
4 Diseño del sistema 15
4.1 Slack 15
4.2 VPS 16
4.3 Servicio Web 16
4.4 Slack Bot 17
4.5 Peticiones API 17
4.6 Actualizaciones 20
4.7 Votaciones 21
4.8 Mensajes interactivos 22
5 Desarrollo 25
5.1 Configuraciones 25
5.2 Slack 25
5.3 Bot y el usuario 27
5.4 VPS 28
5.4.1 Webservice 29
5.4.2 Desarrollador y VPS 31
6 Integración, pruebas y resultados 33
6.1 ChatBot con Socio 33
6.2 Chatbot con Slack 34
6.3 Chatbot con Webhooks 34
6.4 Chatbot con Mysql 35
6.5 Pruebas 35
7 Conclusiones y trabajo futuro 39
7.1 Conclusiones 39
7.2 Trabajo futuro 39
Referencias 40
Glosario 41

INDICE DE FIGURAS

Ilustración 1 – Búsquedas de la palabra chatbot en Google
Ilustración 2 – Vista de un canal en Slack
Ilustración 3 – Mensaje con opciones Ilustración 3 – Mensaje informativo
Ilustración 4 - Diseño global del sistema
Ilustración 5 - Arquitectura para bots de Slack
Ilustración 6 - Diagrama de funcionalidad del Slackbot
Ilustración 7 - Proceso petición a Socio
Ilustración 8 - Proceso de actualización
Ilustración 9 – Lanzamiento de consenso
Ilustración 10 - Proceso de votación
Ilustración 11 - Proceso mensaje informativo
Ilustración 12 - Proceso mensaje interactivo
Ilustración 13 – Imagen al añadir un elemento al proyecto
Ilustración 14 – Imagen al eliminar un elemento
Ilustración 15 - Integración en VPS
Ilustración 16 – Debug de Flask
Ilustración 17 – Ejemplo de devolución de SOCIO
Ilustración 18 – Ejemplo información de error
Ilustración 19 – Gestor de errores
Ilustración 20 – Proceso de crear un proyecto
Ilustración 21 – Proceso de obtener estadísticas

INDICE DE TABLAS

Tabla 1 - Configuraciones de los elementos

1 Introducción

En este primer aparatado vamos a destacar las motivaciones e intereses que nos llevan a la realización del proyecto. Estas motivaciones nos ayudan a centrarnos en un entorno más concreto, ya que hoy en día, el mundo de las tecnologías abarca un montón de temas e ideas a desarrollar. En esto consiste el siguiente apartado, se analizarán cuáles serán los principales intereses para la realización de este proyecto.

1.1 Motivación

La realización de este proyecto viene fomentada por la oportunidad de implantar un software de modelado para redes sociales en una de las redes de carácter más empresarial, Slack, a través de un chatbot, ya que a pesar de su gran cantidad de herramientas Slack no posee ninguna de modelado.

Hoy en día, la automatización de tareas es un punto clave para la gestión, desarrollo y mantenimiento de cualquier sistema o infraestructura informática. El desarrollo de bots favorecen la automatización de tareas programadas, esto aporta mucho peso los bots a la hora de customizar recursos y procesos que hasta el momento se hacían manualmente, como puede ser, dar de alta a un usuario en un sistema o el envió de un correo electrónico.

La finalidad de un chatbot es la interacción con usuarios y realizar las acciones que este le diga, esto es una funcionalidad que se adecua perfectamente para implantarse a la infraestructura que va a existir entre la aplicación de modelado y Slack.

Uno de los atractivos de las redes sociales son la usabilidad donde interfieren tanto en el ámbito empresarial como personal. Hay muy pocas marcas comerciales, empresas y personas que no usen este tipo de redes.[22]

El uso de este tipo de redes, donde destaca la interconexión y relaciones entre personas es ideal para explotar un software de modelado de proyecto, pudiendo fácilmente interactuar con los compañeros de proyecto.

Una herramienta de modelado es una atractiva elección junto con Slack, y chatbots para la creación y explotación de ideas que están en pleno auge.

Más adelante se hablará de que es Slack, pero haciendo un breve resumen introductorio podemos mencionar que es una aplicación de mensajería instantánea para grupos de trabajo que es muy usada en terrenos empresariales y que esta teniendo una gran importancia en el panorama actual.

1.2 Objetivos

El objetivo principal de este proyecto es crear un chatbot para modelado colaborativo en Slack. Esté chatbot permitirá la comunicación entre los usuarios de Slack y una API, SOCIO [2] que procesa el lenguaje natural y crea modelos de dominio.

En todo este entramado de desarrollo y comunicaciones entran en juego diferentes elementos de los cuales, o desarrollaremos o tendremos que documentarnos. Será necesario para llevar a cabo las tareas de creación del bot, como pueden ser, comunicaciones entre usuario de Slack y el bot, viceversa y sus correspondientes respuestas.

El bot proporcionará respaldo de la API SOCIO a Slack, en la cual se podrán ir gestionando, visualizando, visualizar historiales y hasta realizar consensos de posibles cambios.

La comunicación entre la API, y el usuario final de Slack será el objetivo del bot, haciendo que el usuario escribiendo un comando e interactuando con el bot pueda ir llevando a cabo los diferentes desempeños de la aplicación, que, a grandes rasgos, permite realizar proyectos de modelado de forma colaborativa. Además, proporciona la funcionalidad de gestión de proyectos, así como la capacidad de crear varias ramas a raíz de un proyecto, pudiendo, más tarde, seleccionar alguna para integrarse con el proyecto raíz.

1.3 Organización de la Memoria

La memoria consta de los siguientes capítulos:

- Componentes, tecnologías que intervienen en el proyecto: se pondrá en conocimiento uno a uno los elementos que forman el sistema.

- Diseño, desarrollo e integración de los sistemas: podremos ver como se ha tenido que plantear y llevar a cabo la realización del proyecto para su total funcionalidad en tiempo real.

- Pruebas, resultados y análisis: veremos que acciones hemos tenido en cuenta para el correcto funcionamiento del bot.

- Desarrollos futuros: futuras ideas reales que se pueden llevar a cabo.

2 Estado del arte

En esta sección hablaremos fundamentalmente de como se encuentran hoy en día las tecnologías, métodos o sistemas de la cual parte este proyecto. Como es sabido este proyecto trata sobre la realización de un chatbot sobre Slack para una Api de modelado, sabiendo esto, podremos ver como se encuentra en la actualidad el tema de los chatbots a nivel global.

También se verá un breve análisis sobre los bots que se encuentran en Slack y de otras aplicaciones de modelado que están hoy en día disponible. Por último, estacaremos que funcionalidades o que recursos aporta este proyecto en la actualidad y en que se puede diferenciar del resto de sus competidores.

2.1 Chatbots

El origen de la palabra chatbot provine de un juego de mazmorras que se llamaba chatterbot[23], la lógica del juego era responder preguntas que desencadenaban unas acciones, pero es el origen semántico. El primer bot se creó llamado Eliza, en 1964, por un desarrollador llamado Joseph Weizenbaum del MIT partiendo de la idea de Alan Turing con las maquinas de estado. Eliza [3] era capaz de conversar con las personas haciéndolas creer que estaban hablando con otra persona.

La interacción es sencilla, un usuario interactúa con ellos, estos analizan la cadena de texto o voz introducida por la persona y les responden según se haya programado. Puede haber bots que se basen en el uso de reglas por medio de comandos, pero hoy en día podemos encontrar chatbots con inteligencia artificial con capacidades de aprendizaje.

Los chatbots están muy metidos en nuestro día a día, nos lo podemos encontrar cuando hacemos alguna compra en alguna plataforma online, atención al cliente, y en comunicaciones y redes sociales.

Las empresas en el ámbito financiero y tecnológico piensan en el bot como un recurso económico y siempre para la gestión de tareas y la distribución de información tanto a nivel de marketing como a nivel de comunicación directa al trato con sus clientes reduciendo enormemente las horas de trabajo y recursos humanos. [4]

El auge de estos puede verse reflejado en la búsqueda realizas a lo largo de 5 años atrás en el motor de búsqueda Google [5]. En la ilustración 1 puede verse el gráfico de búsquedas de la palabra chatbot donde se ve a lo largo de 5 años cuantas veces al día se ha realizado una búsqueda con esa palabra:

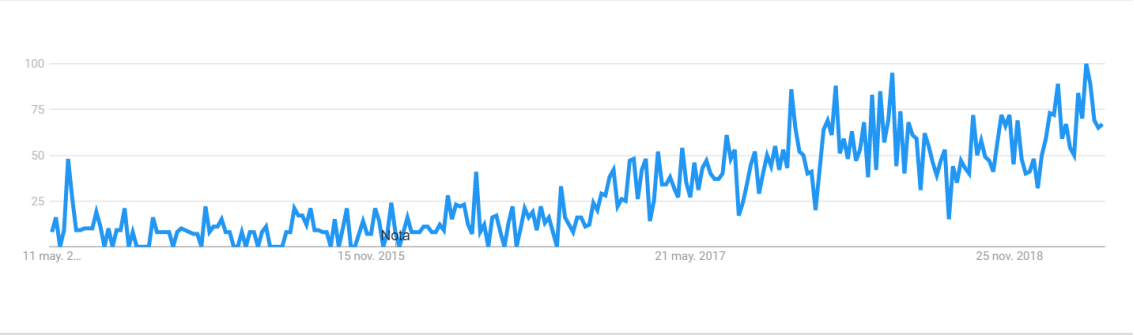


Ilustración 1 – Búsquedas de la palabra chatbot en Google

Como se puede observar con el paso de los años ha ido creciendo el numero de búsquedas, llegando a finales de 2018 con el máximo alcanzado hasta la fecha, esto conlleva a pensar que es un término que, al estar ligado a las tecnologías, el desarrollo de estos está creciendo.

Una de las razones por lo que la terminología chatbot se le ha dado a conocer, es por su intrusión en el mundo de las redes sociales.

Es común encontrarse bots en las redes sociales, capaces de gestionar y reproducir o interactuar con herramientas y programas en la nube, pudiendo interactuar con los usuarios de manera casi imperceptible de que es un software, como puede ser TIDI [7] asistente del comercio electrónico udespensa.com que sirve para buscar y pedir los productos que tienen en la tienda o Julia[6] un asistente virtual para banca, con respuestas personalizables, es capaz de contestar 900.000 preguntas relacionadas con acciones, fondos, bonos...

Una de las mejores plataformas de comunicación donde se alojan bots con una gran productividad es Slack. En ella se pueden encontrar multitud de bots los cuales pueden despeñar varias funciones, desde análisis sintácticos como pueden ser Codeship[8], automatizar mensajes grupales integrados en aplicaciones de desarrollo de código, como pueden ser obie[9], jira cloud[10] herramientas para la administración de proyectos, seguimientos de posibles errores.

En el ámbito en el que se va a desarrollar el bot no se encuentran bots parecidos a SOCIO, nos tenemos que ir a Zepelin[12], la cual se basa en la colaboración para la creación de guías de estilo, o github[13] para desarrolladores de código, pero no a nivel de gestión de modelos de proyecto, esto representa una gran ventaja a la hora de implantar un bot de estas características en esta plataforma.

Aunque pueda haber otros dos bots que interaccionan con SOCIO [2] y se integran en otras redes sociales (Twitter y Telegram), Slack es una red orientada a los entornos de trabajo y a mensajería directa colaborativa son dos pilares que hacen de este bot más atractivo que sus competidores.

2.2 Aplicaciones de modelado y aplicaciones colaborativas

La cooperación en los proyectos de desarrollo de software fomenta la continua comunicación, esto es debido a que los proyectos se gestionan en paralelo desarrollándose módulos por separado. Hoy en día existen herramientas online colaborativas donde se pueden cooperar de manera síncrona, esto significa que las modificaciones se realizan en todos los participantes de ese proyecto, esto le da una ventaja con herramientas que son asíncronas, facilitando la rapidez y gestión de cambios en los proyectos.

Algunas herramientas que podemos encontrar son:

- GenMyModel [14]: empezó como una herramienta sólo para UML. Pero desde entonces se ha ido expandiendo hasta cubrir áreas complementarias como seria el modelado de procesos de negocio y el modelado de la arquitectura empresarial. Además, ofrece un repositorio centralizado para los equipos (con posibilidad de definir políticas de restricción de acceso) para el modelado colaborativo simultaneo.

- Gliffy [15]: es una de las herramientas más popular para la creación de diagramas online, permite modelar todos los diagramas UML, así como una variedad de otros tipos de diagramas. Además, vienen con plugins, muy útil si se necesita las actividades de modelado con el resto de las etapas del proceso de desarrollo.

- Diagramo [16]: herramienta gratuita que se puede instalar en tu propio servidor. Su pega es que está muy enfocada al modelado de diagramas de flujo con lo que, a nivel de UML, sólo permite el modelado de máquinas de estado. La colaboración síncrona no es posible.

Hoy en día el trabajo colaborativo hace que muchas tecnologías estén desarrollándose en la nube y que a la vez sea accesible y editable al mismo tiempo por varias personas. Desde Google con su editor de documentos incorporado en Google Drive [17], como office 365[18], la compartición de archivo como Dropbox[19] o WeTransfer[20] el cual se puede enviar documentos a través de Emails.

3 Componentes del sistema

El proyecto consta de diferentes elementos que hacen los pilares del mismo, todos y cada uno de ellos contribuyen a la base de conocimiento para poder realizarlo. La mejor manera posible es poder ir viendo cada uno de ellos individualmente, conociéndolo qué son, que funciones tienen tanto a nivel de proyecto como de desarrollo.

Estos componentes que hacen que puede existir un intercambio de recursos, información y respuestas a tiempo real. Unos se encargan de entrelazar comunicaciones, otros albergan software y otros son el núcleo del proyecto, pero es necesaria saber que son y que hacen.

3.1 API MODELADO

En primer lugar, vamos a describir lo que es una API (Application Programming Interface). Una API es un conjunto de procedimientos que cierto software dispone para poder comunicarse con otros sistemas. Hay varias maneras en las cuales dos sistemas puedan comunicarse, y podemos diferenciar algunos grupos habituales de API's que existen actualmente:

- Servicios web: intercambios entre sistemas o aplicaciones que traspasan información vía HTTP generalmente o HTTP's en caso de facilitar o generalas los certificados pertinentes. El intercambio de información suele producirse a través de tipos de formato JSON y XML. El tipo de api de SOCIO, API con la cual es la que se ha trabajado, ha sido REST (**R**epresentational **S**tate **T**ransfer). Se apoya en llamadas como peticiones vía HTTP.

- Bibliotecas: permite que un sistema importe una biblioteca como intercambio de información. Un ejemplo puede ser Google Maps como api orientada a bibliotecas.

- Sistema Operativo: los propios softwares interaccionan con los sistemas operativos en muchos casos, esas comunicaciones con el sistema se producen gracias a las APIS de los propios sistemas.

Una vez conocido que es una API podemos explicar cómo es SOCIO la API de modelado que se ha usado para generar el BOT en SLACK.

SOCIO es el sistema encargado de gestionar y los modelos de un proyecto. Esta API de tipo REST recoge las peticiones que se envían vía HTTP para realizar las acciones pertinentes. La funcionalidad de esta nace en la necesidad de poder compartir información acerca de un proyecto de desarrollo. Es sabido que en el desarrollo del software intervienen varias personas encargadas de diferentes tareas, por tanto, la necesidad de comunicación entre ellas es bastante alta, además en muchas ocasiones el desarrollo de funcionalidades va en paralelo y es necesario conocer ciertos códigos de otros desarrolladores. Aquí entra en juego SOCIO, con la cual podemos ir gestionando proyectos, consultando información

de cómo van ciertos desarrollos, creaciones de Branch para futuros consensos con los desarrolladores.

SOCIO guarda los modelos en proyectos. Para restringir la accesibilidad de los proyectos, estos pueden ser públicos, todo el mundo puede leer y escribir en un proyecto, privados, solo por invitación se puede leer y escribir y protegidos donde los usuarios pueden leer, pero necesitan permiso de escritura. Además, procesa el lenguaje natural interpretándolo para crear los modelos.

Otra de las grandes características de socio es la manera de realizar los modelos. Los usuarios crean en lenguaje natural los modelos, SOCIO los interpreta gracias al analizador sintáctico The Standford Parser[21], que separa y analiza cada frase y crea el árbol sintáctico, Funciona de tal manera que se crean reglas para las frases a analizar, y realiza las acciones pertinentes según se haya buscado el modelo, se gestionan los cambios internamente y gracias a PlantUML que genera imágenes con el estado del modelo, estas son devueltas al usuario para que pueda comprobar la acción que ha realizado sobre el proyecto.

Además de la interacción con usuarios de diferentes redes sociales, creación de modelos a través de lenguaje natural y de la creación de estadísticas de los mismos, SOCIO puede generar consensos entre los usuarios sobre diferentes versiones de proyectos que han ido creando sobre uno mismo, Branch, para saber qué forma tiene que llevar el proyecto y estos votar o decidir cuál es el correcto.

Su funcionalidad no se resume en tres acciones, si no que podemos:

- Gestión de proyectos:
    - Crear Proyecto
    - Eliminar Proyecto
    - Cambiar la visibilidad (Public, Private, Protected)
    - Validar proyecto
    - Dar permisos a un usuario
    - Quitar permisos a un usuario
    - Cambiar permisos a un usuario
    - Crear un Branch
    - Cambiar el grupo del Branch

- Acciones de modelado:
    - Realizar acción modelado
    - Deshacer acción modelado
    - Rehacer acción modelado

- Obtener información sobre:
    - Todos los proyectos
    - Proyectos de un usuario
    - Proyectos en los que un usuario puede escribir
    - Proyectos en los que un usuario puede leer
    - Proyectos con cierta visibilidad
    - Proyecto concreto

- Estadísticas:
    - Todas las acciones realizadas en un proyecto
    - Acciones por usuario, donde las acciones son los cambios de modelado que ha hecho sobre cierto proyecto
    - Mensajes por usuario
    - Porcentaje de auditoria

- Historial:
    - Acciones de modelado
    - Acciones de gestión de proyecto
- Obtener el modelo
- Obtener ayuda

- Votaciones:
    - Seleccionar Branch
    - Empezar proceso de consenso
    - Empezar ronda de votación

Estas son todas las peticiones que se pueden realizar a la API, aunque el verdadero potencial reside en la creación de modelos de los proyectos. SOCIO contaba ya con Telegram y Twitter, puede observarse la compatibilidad que tiene un api de poder gestionar las comunicaciones con sistemas y plataformas diferentes.

Como se puede observar, SOCIO es una pieza clave del proyecto ya que el bot creado le dará soporte a SOCIO y en este caso a la red social Slack de la que hablaremos a continuación.

3.2 SLACK

Hoy en día las redes están teniendo una gran importancia en el mundo de las tecnologías, una de las aplicaciones o herramientas que está teniendo un gran auge en grupos de trabajo es Slack, facilita la comunicación a tiempo real en grupos de trabajo. Podemos crear canales donde interactuar con los usuarios, gestionar los proyectos y recibir las notificaciones. Además, destaca la capacidad de integración de aplicaciones cómo Twitter, Google Drive, Dropbox, Bitbucket…

Otra de sus características, es la que hace posible la creación de bots para poder gestionar y automatizar ciertas acciones, interacciones con el usuario y/o aplicaciones. Slack es una red de mensajería que se centra en entornos empresariales, además al tener la capacidad de integración y desarrollo de aplicaciones de terceros y bots, es muy usada por entornos tecnológicos y grupos de desarrollo.

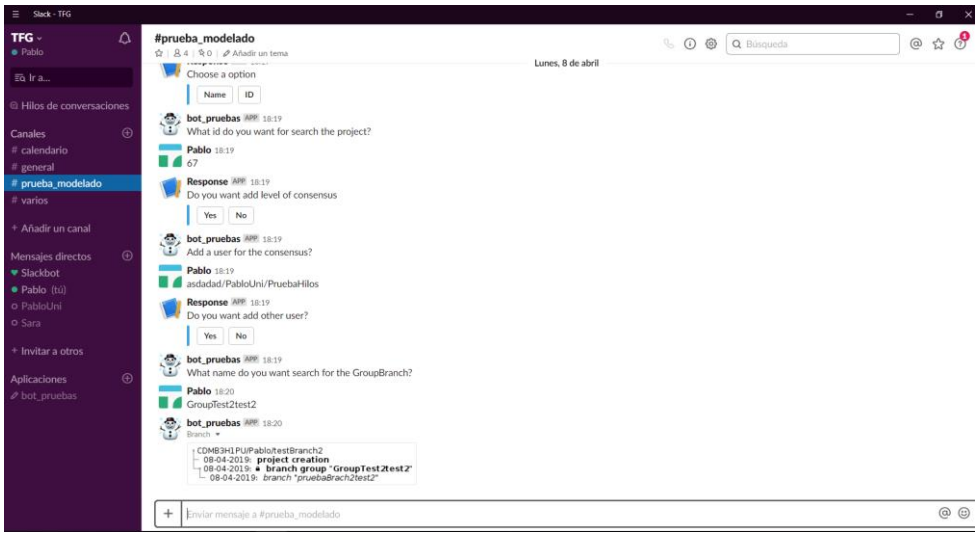


Ilustración 2 – Vista de un canal en Slack

En la ilustración 3 podemos observar como se ve la interfaz grafica de Slack sobre la APP de Windows 10.

Todo esto hace que Slack y SOCIO encajen a la perfección siendo el bot que trata el proyecto la conexión con ambos. Al poder recibir imágenes, SOCIO puede enviar los estados y actualizaciones de los proyectos a los diferentes canales creados, al igual que recibir la información.

3.3 VPS

Un VPS (servidor virtual privado) es una partición en un servidor físico. Esto hace que cada partición pueda virtualizar sus recursos. En él se pueden instalar el sistema operativo que se desee.

Se usa una capa de virtualización sobre el sistema operativo del servidor, pudiendo particionar y emular diferentes sistemas operativos en el mismo servidor físico. Además, también se pueden asignar los recursos dependiendo de las necesidades de cada usuario.

Los encargados de esta tarea de virtualización, gestión de recursos y mantenimiento de los servidores físicos se realiza por medio de proveedores de servicios. Para el proyecto se ha elegido OVH, que ofrece una gran cantidad de servicios hosting. Las características del VPS seleccionado han sido muy básicas:

- KVM OpenStack
- 1 core 2 Gh
- 2 Gb Ram
- 20 Gb ssd

Una vez contratado el VPS, lo primero que el proveedor de hosting nos facilita es la ip y una contraseña para la conexión al mismo. Esta se realiza gracias al protocolo SSH que usa el puerto 22 para la conexión.

3.4 Botones interactivos

Hay otro elemento en el proyecto que hay que resaltar que ha hecho posible que la interacción con el usuario fuera más intuitiva y versátil. Para poder crear elementos interactivos como botones donde el usuario puede elegir o responder a las preguntas del bot ha sido necesaria añadir una aplicación de terceros de Slack, Webhooks, que funciona junto con la api y hace posible crear esos elementos.

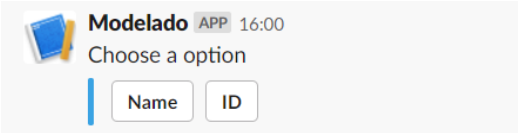


Ilustración 3 – Mensaje con opciones

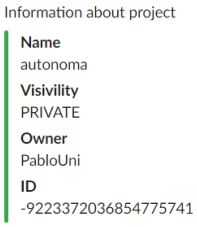


Ilustración 3 – Mensaje informativo

En la ilustración 3, izquierda, podemos ver una representacion de como se van a presentar los mensajes que requieran. Al lado tenemos como se mostrará la información que solicitemos.

3.5 Chatbots

Ya hemos mencionado anteriormente que eran este tipo de tecnología. En la actualidad se pueden encontrar en muchas plataformas y servicios informáticos. Las empresas están implantando esta tecnología para la gestión y el trato con personas y gracias a los estudios y desarrollos con la IA estos están adquiriendo una importante relevancia en el futuro de las comunicaciones.

Si nos adentramos en cualquier plataforma de comercio online es raro no encontrar una atención al cliente, gestionada por chatbots, estos además de responder con información necesaria son capaces de realizar ciertas tareas. Como se ha mencionado antes, la IA funcionando con chatbots, pueden alcanzar conversaciones fluidas y atender perfectamente al cliente si necesidad de tener una persona insitu.

No solo los bots están centrados en el ámbito de las interacciones humanas, sino que son perfectos autómatas a la hora de la gestión de tareas aumentando la productividad debido a que están programados para tareas específicas siendo unos autómatas que se adaptan a las posibles situaciones que pueden surgir en el desempeño de su función.

Otro de los usos menos técnicos que pueden darse en las redes sociales, donde estas integran estos componentes pudiendo hacer de ellos nuestros pequeños gestores en las diferentes redes sociales, haciéndonos de filtro o incluso responder ciertos tipos de mensajes en nuestro lugar.

Ejemplos claros tenemos a Google y Apple con sus asistentes para servicios. En Slack también hay una gran variedad de bots ya implementado que van desde la visualización del estado de envió de tus paquetes, un filtrador de mensajes, pasando por un gestor de gastos.

En este proyecto esta es la pieza principal, sobre él giran todos los componentes y es el motor del proyecto. Se encarga del trato con el usuario, así como de las repuestas interactivas, de realizar las respectivas gestiones y notificárselo a la aplicación de modelado, recibir la información que el usuario ha solicitado y devolvérsela.

Además, no se ha encontrado ningún Bot que gestione modelos de domino, esto hace de este Bot un atractivo instrumento para Slack y la fuerte repercusión que puede tomar Socio en esta plataforma.

4 Diseño del sistema

Una vez conocido todos los subsistemas que conforman el proyecto podemos dar una versión global del diseño que conforman cada uno de los elementos. En la imagen 4 se puede observar cómo es el diseño del sistema y las comunicaciones que existen entre los diferentes componentes.

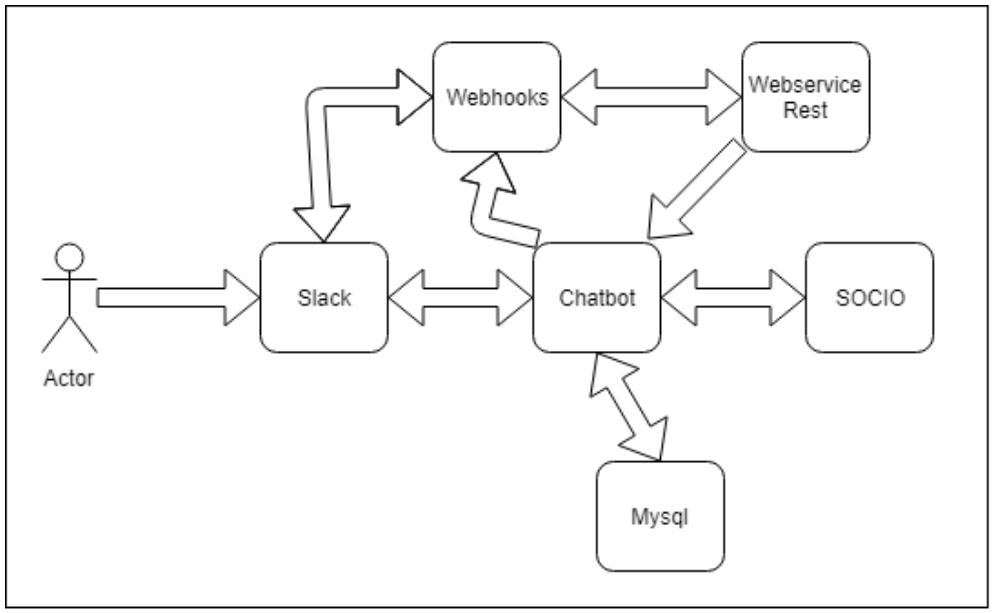


Ilustración 4 - Diseño global del sistema

Tomando como idea la forma que va a tener el sistema, sus componentes y las comunicaciones que hay entre ellos, podemos fraccionar y orientar los diseños y desarrollos de cada uno de los mismos.

4.1 Slack

El sistema de comunicación de Slack se realiza a través de canales, ya sean privados o grupales. Esto, se debe tener en cuenta a la hora de desarrollar el chatbot ya que habrá que hacer distinciones entre canales y la privacidad de estos. Antes de la creación de canales, Slack aglutina esos canales en espacios de trabajo, donde en cada espacio de trabajo podremos agregar, eliminar y gestionar los diferentes canales, aplicaciones y bots que actuaran.

Slack hace uso de aplicaciones, herramientas de terceros, que se pueden integrar en el espacio de trabajo y estas ayudarnos a realizar ciertas tareas. La orientación que da Slack a los bots es parecida a una aplicación, la cual se crea y se integrara en el espacio de trabajo y a su vez albergara el chatbot.

Una vez creada la aplicación, podremos asignar a esta un usuario de tipo bot. Esto hará que cuando se añada esa aplicación al espacio de trabajo podrá contar con un usuario bot, que en este caso será nuestro chatbot que hará de intermediario entre SOCIO y Slack.

Se puede apreciar que el sistema de gestión de bots para Slack puede ser un poco confuso, pero aporta un nivel de seguridad, integración con el sistema bastante fiable.

Los párrafos anteriores hacen referencia la imagen 5, donde se podrá visualizar la arquitectura y el encapsulamiento que hace Slack con los bots para la gestión de canales.

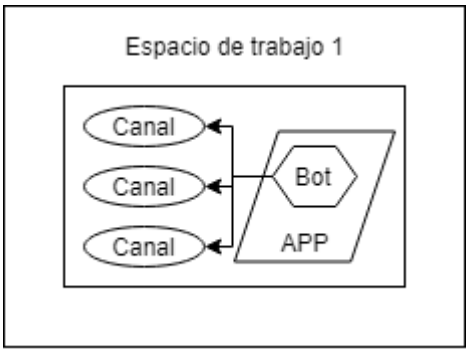


Ilustración 5 - Arquitectura para bots de Slack

Este bot será único, ya que Slack nos genera un Token para su uso y control. Este token será necesario para el desarrollo e integración del bot con Slack.

El usuario podrá conversar con el bot, haciéndole llegar las órdenes a SOCIO y este responder con la información correspondiente.

El bot será el encargado de enviar la petición a la API de Webhooks y este se encargará de mostrarlo en el canal correspondiente en Slack, según se haya configurado la app.

4.2 VPS

Para alojar el servicio web y poder obtener una URL con una ruta estática, donde la IP publica no cambie, una posible solución ha sido contratar servicios de hosting. Si se quisiese alojar un host de manera local puede ser que la IP publica pudiera verse afectada y tener que volver a configurar Webhooks. Por esto, será necesaria la creación de un entorno en la nube capaz de alojar 24/7 los servicios necesarios para el mantenimiento e implantación de este subsistema.

4.3 Servicio Web

Para poder recoger las respuestas de los usuarios, ha nacido la necesidad de crear un webservice. Por norma Webhooks tiene por defecto el envió de la información contenido en un archivo JSON, una de las posibilidades para poder gestionar esta comunicación fue el uso de un servicio web rest.

Este servicio web se alojará en el VPS y esperará las respuestas del usuario para hacérsela llegar al bot.

4.4 Slack Bot

Este es el pilar del proyecto, es el encargado de establecer la comunicación entre los usuarios de Slack y la API de modelado. La plataforma Slack facilita a través de las aplicaciones la gestión de los bots.

El bot siempre estará escuchando las posibles llamadas que los usuarios puedan realizarle, siempre con una @botModelado y cuando en el canal donde se le quiera llamar este integrada la aplicación que lo gestionará tanto Webhooks para su correcto e integro funcionamiento.

Además de escuchar las peticiones de los usuarios, el bot será capaz a través de la gestión de hilos estar pendiente de las posibles modificaciones de un proyecto, en el cual un usuario de Slack estuviera implicado en el. Así como si se lanza una votación para determinar que branch es el seleccionado por la gente. De tal modo, este bot estará continuamente escuchando las peticiones de ambos lados de la conexión.

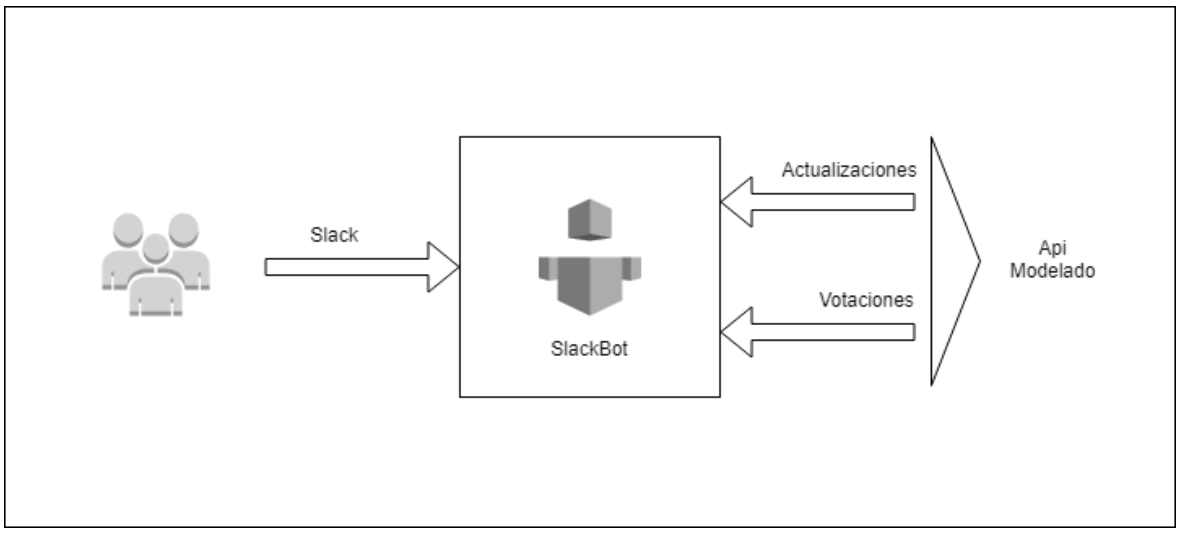


Ilustración 6 - Diagrama de funcionalidad del Slackbot

La figura 6 muestra como el bot es capaz de estar escuchando las peticiones en tres frentes diferentes.

4.5 Peticiones API

La funcionalidad principal del chatbot que se ha diseñado es la de intercomunicar Slack con la API de modelado. Como se ha mencionado anteriormente para comunicarse con el bot es necesario escribir @botModedo y tras esto el comando que queramos realizar. Los posibles comandos son:

- help: muestra al usuario la lista de comandos posible
- createproject: crea un proyecto
- getprojects: obtiene los proyectos del propio usuario
- getprojectsuser: obtienes los proyectos de un usuario
- getprojectsuserwrite: obtienes los proyectos de un usuario en los que tiene permiso de escritura
- getprojectsuserread: obtienes los proyectos de un usuario en los que tiene permiso de lectura
- getprojectvisi: obtienes los proyectos con cierta visibilidad
- getinfoproject: obtienes información de un proyecto en concreto
- deleteproject: borras un proyecto
- changevisibility: cambias la visibilidad a un proyecto
- validateproject: valida un proyecto
- adduser: añade un usuario a un proyecto
- deleteuser: borra un usuario
- updateuser: actualiza los permisos a un usuario
- createbranch: crear un Branch
- groupbranch: añadir grupo al Branch
- getstatistics: obtener las estadísticas de un proyecto
- getstatisticsbyuser: obtener las estadísticas de un usuario
- getmodel: obtener un modelo
- historyproject: historial de un proyecto
- doaction: realizar una acción
- undoaction: deshacer una acción
- redoaction: rehacer una acción

- getlastupdate: obtener la última información de un proyecto
- selectbranch: seleccionar un branch
- startconsensus: empezar un consenso

La lista de comandos viene especificada en lengua inglesa, por lo que la comunicación con el bot será de la misma manera.

Para otorgar una comunicación entre el usuario y el bot bastante sencilla se ha dispuesto de una lógica de comunicación de preguntas. Una vez que el bot recibe el comando del usuario este empieza a realizar las pertinentes preguntar para ir completando la información necesaria para que, al enviar la petición a la ruta, contenga la información precisa y la devolución de la API sea la correcta.

Así mismo, entre la Api de modelado y el chatbot se han tratado todas las devoluciones que la Api devolvía en caso de error, gestionando así posibles errores en las comunicaciones o procesos de información y comunicándoselo al usuario sin necesidad de parón de servicios.

- 500: errores internos al cual el usuario se les notifica como “recurso no disponible”
- 403: dependiendo el tipo de petición que se había realizado, el proyecto o Branch no existía o el usuario que se solicitaba la información no tenía permisos en el mismo. Estos mensajes de la Api se les devuelve el formato de cadena de texto.
- 200: son mensajes los cuales han salido correctamente la petición, dependiendo del tipo de petición que se haya realizado devolveremos una imagen o cadena de texto un mensaje interactivo con la información que se ha solicitado. Asimismo, se ha tenido que hacer un tratamiento de las imágenes para que puedan ser visibles por el usuario a través de Slack.

Como se puede apreciar en este apartado, el bot realizara una labor de recolección de datos dependiendo del tipo de comando que le haya solicitado el usuario.

En la siguiente imagen se puede apreciar el proceso en el que el bot prepara correctamente la petición para la Api:

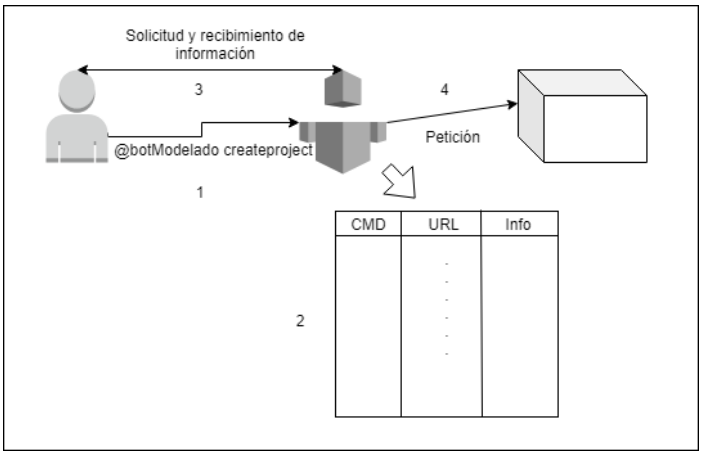


Ilustración 7 - Proceso petición a Socio

1. El usuario lanza un comando, en este caso crear un proyecto.
2. El bot ve que comando le ha llegado, el cual tiene asociado una URL para el envío de la petición a SOCIO y una serie de parámetros los cuales se corresponden a la información necesaria para poder efectuar correctamente la petición.
3. Solicitamos y recopilamos toda la información necesaria.
4. Se envía la petición a SOCIO.

Una vez que la petición se ha realizado, la Api de modelado nos puede devolver diferente tipo de información como se ha visto unos párrafos antes: una cadena de texto o una imagen. Todos estos tipos de información también tiene que ser procesada por el bot, además, de los mensajes interactivos para mostrar información acerca proyectos y Branchs.

4.6 Actualizaciones

Otra de las funcionalidades del bot aparte de la escucha de peticiones de los usuarios, es la de estar pendiente de las actualizaciones que pueda sufrir un proyecto en cualquier momento.

Cuando algún proyecto ha sufrido algún cambio, y es relevante para alguno de los usuarios que se encuentran en su plataforma este se encargara de enviar una imagen con la modificación que se haya producido.

El funcionamiento de este mecanismo es relativamente sencillo, el bot lanzara a través de un bucle de programación la misma petición continuamente. Esta petición de tipo GET se encarga de ver si hay alguna actualización, devolviendo siempre un array de proyectos actualizados vacíos. En el momento en que esta petición devuelva un array no vacío, que contendrá el proyecto actualizado vera quienes tienen permiso de lectura, se obtendrá la última imagen de la actualización y se le enviara a cada usuario a través de su canal correspondiente.

Todo este proceso puede verse simplificado en la siguiente imagen:

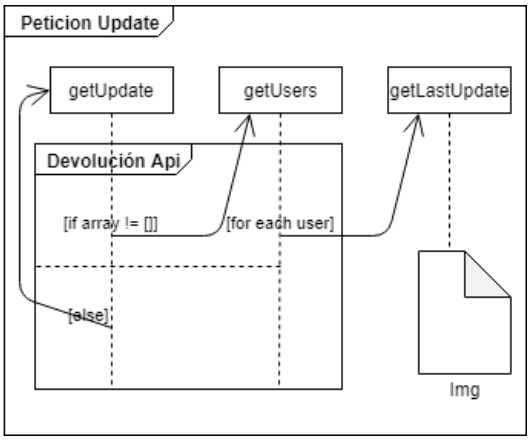


Ilustración 8 - Proceso de actualización

4.7 Votaciones

El sistema de votaciones se implanta paralelamente al de la escucha de peticiones de usuarios de Slack, como las peticiones que se lanzan para ver las posibles actualizaciones a tiempo real. Estas tres funcionalidades se gestionan con hilos y el hilo que gestiona las votaciones es muy parecido al hilo que gestiona las actualizaciones.

De la misma manera, se lanza dos peticiones en dos hilos diferentes, una para ver si una votación ha finalizado y enviar así la información con la opción vencedora ha salido elegida, como así mismo una en la que escucha si se ha enviado algún consenso.

Cuando se lanza un consenso, el cual implica automáticamente una votación o varias en caso de no llegar al consenso, el usuario que envía el consenso tiene la capacidad de dar las opciones de establecer las opciones de votación, una vez lanzada la petición obtendremos la jerarquía del proyecto:

Ilustración 9 – Lanzamiento de consenso

Cuando se recibe una votación, el bot a través del mismo proceso de actualización, recoge la lista de usuarios de Slack que deben botar. Una vez seleccionado los usuarios, se tienen que obtener las opciones a elegir, para esto, se obtiene los branch abiertos, se obtienen sus imágenes correspondientes con su última modificación, tras esto, se les envía como mensaje privado a cada usuario para que pueden elegir que opciones prefieren.

Una vez los usuarios han elegido sus preferencias, el bot envía sus votos a la Api de modelado y tras un periodo de tiempo esta le devuelve el resultado de las mismas para que el bot se le haga llegar a los usuarios.

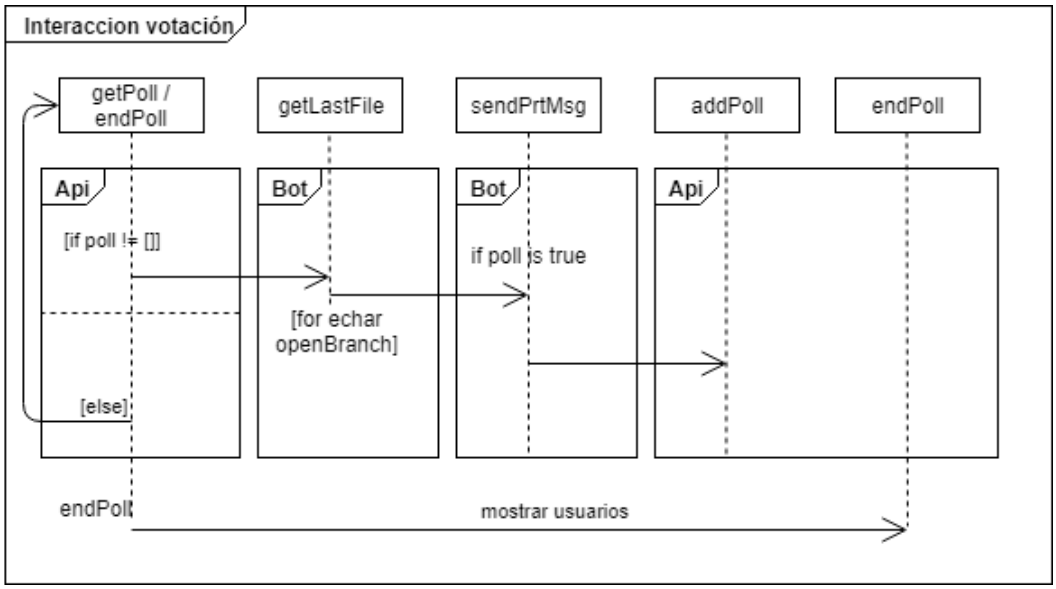


Ilustración 10 - Proceso de votación

En la figura 10 podemos ver todo el proceso que se lleva a cabo:

- Recibir votación
- Recoger cada uno de los modelos de los proyectos
- Enviarle esos modelos por mensaje privado a cada uno de los usuarios para que hagan sus elecciones
- Añadir las votaciones al consenso
- Mostrar el resultado a los usuarios

4.8 Mensajes interactivos

Como hemos mencionado anteriormente, apartado 3.1, Slack incorpora aplicaciones de terceros para poder ampliar la funcionalidad y mejorar la plataforma. Una de estas aplicaciones es Webhooks usada en el proyecto.

Gracias a esta aplicación que se integra juntamente con la aplicación que contiene al bot podemos mandar mensajes de tipo interactivos, ya sean botones, formularios o mensajes de aviso. Es una manera de tener un dialogo con el usuario más efectiva, ya que al enviar botones o checkOptions de respuesta simplificando la acción con un click

El funcionamiento de Webhooks puede resultar un poco complicado e implica la necesidad de desarrollar un servicio web para poder obtener las respuestas de los usuarios y a u su vez este comunicárselo al bot el cual está esperando la respuesta. Como se ha mencionado antes, Webhooks es otra app que se integra conjuntamente con la aplicación

Se pueden destacar dos servicios que realiza esta app:

- *Envió de mensajes interactivos:* para el envío de este tipo de mensajes y la llegada al usuario final el Bot realiza una petición vía HTTP a la Api de Webhooks. Gracias a la previa configuración de la Webhooks en Slack, nos devolverá la URL donde tenemos que enviar la información.

- *Recepción de mensajes*: cuando el usuario responde a un mensaje interactivo ya sea de botón como de checkbox esta respuesta será enviada a una URL que se ha tenido que configurar en Webhooks de la misma manera que nos da la URL para enviarle el mensaje interactivo, hay que configurar para ver donde se va a enviar la respuesta.

Aquí es donde entra en juego el servicio web y el VPS (Virtual Private Server) donde se alojará. Nacen de la necesidad de poder recibir la contestación del usuario que proviene de Webhooks y trasmitírsela al Bot que está a la espera de esta.

Ya es sabido como se gestiona los mensajes interactivos. En estos sabemos que interfieren 3 componentes, Slack, el Webservice y el Bot. El funcionamiento difiere en el caso de que el mensaje sea informativo o sea interactivo:

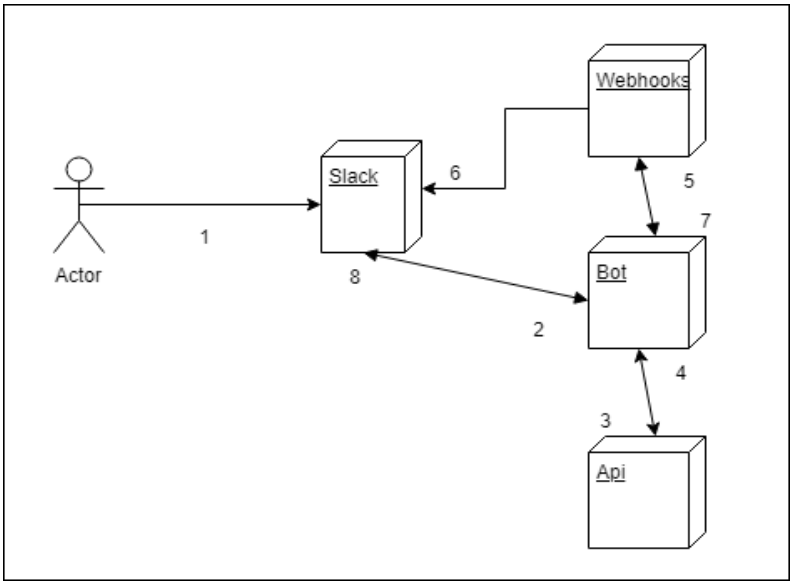


Ilustración 11 - Proceso mensaje informativo

5. El usuario lanza un comando el cual llevara una secuencia de acciones que desencadenara el desenlace de un mensaje interactivo informativo.
6. Slack reconoce que se interacciona con @botModelado y envía la cadena de texto al Bot.
7. Internamente el bot realiza las acciones necesarias y envía la petición a la Api de modelado.
8. La Api devuelve la información correspondiente a la petición recibida.
9. El bot gestiona la petición pertinente con la información recibida de la Api, la envía a Webhooks y espera que el mensaje llegue correctamente.
10. WebHooks envía el mensaje a slack y notifica al bot la llegada del mensaje.

11. El bot envía un mensaje posterior informando del status de la acción.
12. El usuario en cuestión de milisegundos presencia la información solicitada.

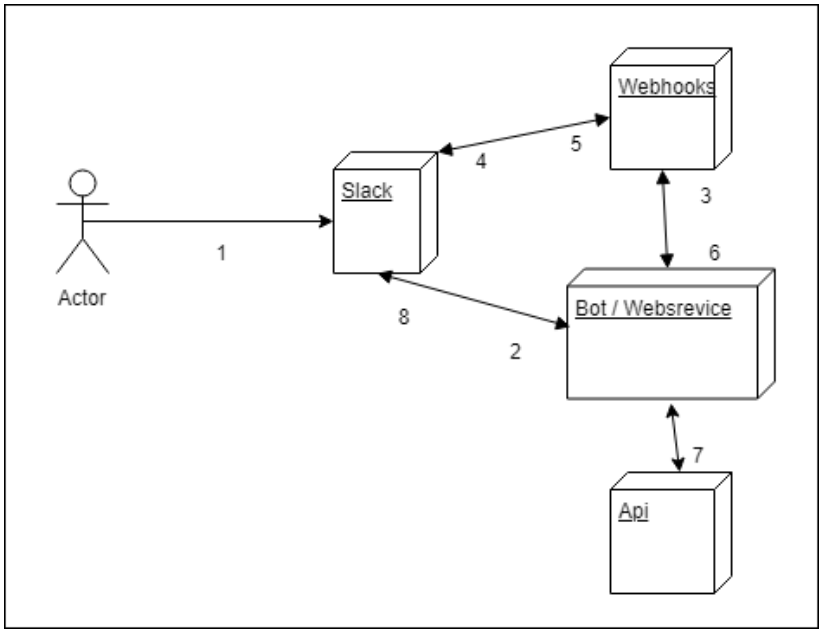


Ilustración 12 - Proceso mensaje interactivo

1. El usuario lanza un comando el cual llevara una secuencia que descaderara las preguntas del bot para cumplimentar la información.
2. Slack reconoce que se interacciona con @botModelado y envía la cadena de texto al Bot.
3. El bot prepara la petición para que Webhooks puede enviar al usuario la pregunta interactiva. Mientras el bot se queda a la escucha de la la información solicitada.
4. A través de la app se le envía al usuario la pregunta para que este responda.
5. Una vez proporcionada la respuesta la aplicación la procesa y envía la información al webservice.
6. El webservice recibe la información y se la hace llegar al bot, el cual, si ha cumplimentado toda la información, enviará la petición o por el caso contrario volverá a seguir realizando preguntas.
7. La Api devuelve la información correspondiente a la petición recibida.
8. El bot procesa la información ya sea una imagen, una cadena de texto o un mensaje informativo, estando esté ligado a realizar los pasos del esquema 1

5 Desarrollo

5.1 Configuraciones

Una de las claves del proyecto ha sido la formación de varios subsistemas y la comunicación entre ambos. Al querer realizar un proyecto e implantarlo de manera real, es necesario para los posteriores apartados otorgar de una identidad a los sistemas, o dicho de otro modo una manera de poder saber a dónde se están realizando las conexiones, esto se realiza a través de IPs y DNS. Para esto se hace referencia en la siguiente tabla.

Sistema	URL	Observaciones
Slack	---	Encapsulamiento gracias a Python de la comunicación
Bot	51.77.145.141	Se alberga en el vps
Api Modelado	http://<servidor>.ii.uam.es	Alojada en servidores de la Autónoma
Webservice	51.77.145.141/slack/message_options	Se alberga en el vps
WebHooks	https://hooks.slack.com/services/<User>/<app>/<Token>	

Tabla 1 - Configuraciones de los elementos

5.2 Slack

Lo primero que hay que hacer en Slack es registrarnos y crearnos una cuenta. Tras esto podremos elegir el tipo de interfaz que deseemos, ya sea web o app para diferentes sistemas operativos y conectarnos.

Para el bot tendremos que ir al panel de configuración que facilita Slack. Como ya se hablo en el apartado 4.1.1 primero se creará la aplicación y tras esto el bot que ira contenido en esa aplicación. La creación del bot generara unas claves necesarias para el desarrollo de ese chatbot.

Una vez creado el bot, podremos integrar la aplicación en el canal y podremos interactuar con el bot.

Otra de las características para el desarrollo son las comunicaciones con Slack, el tipo de formato de mensaje se define gracias a un JSON que se envía juntamente con la petición del mensaje, de esta manera, enviaremos a Webhooks la petición juntamente con el JSON que da formato al mensaje.

Por ejemplo, si queremos enviar un mensaje de tipo botón:

```
json = {
'text': 'Do you want Graphic will be displayed in an absolute or distributed way in the time?',
 'attachments': [{'color': '#3AA3E3', 'text': '', 'actions': [ {'text': 'Yes', 'type': 'button', 'name': 'yes', 'value': 'yes'}, {'text': "No", 'type': 'button', 'name': 'no', 'value': 'no'}],
'callback_id': 'visi_statics',
    'fallback': 'You are unable to choose a option, 'attachment_type': 'default'}]
}
```

Y si lo que queremos es mostrar información:

```
data = {
        "attachments": [{
                "fallback": "Required plain-text summary of the attachment.",
                "color": "#36a64f",
                        "pretext": "Information about project",
                        "fields": [
                                {
                                   "title": "Name",
                                   "value": "%s" % name,
                                   "short": False
                                },
                                {
                                   "title": "Visivility",
                                   "value": "%s" % vis,
                                   "short": False
                                },
                                {
                                   "title": "Owner",
                                   "value": "%s" % user,
                                   "short": False
                                },
                                {
                                   "title": "ID",
                                   "value": "%d" % id,
                                   "short": False
                                }
                            ]
                        }]
        }
```

De esta manera Slack gestiona el tipo y formato de mensajes que se van a presentar en la misma.

Además de mensajes de informativos, también podemos recibir las actualizaciones que ha podido sufrir un proyecto, pero en este caso lo que se envía es una imagen.

- Añadir:

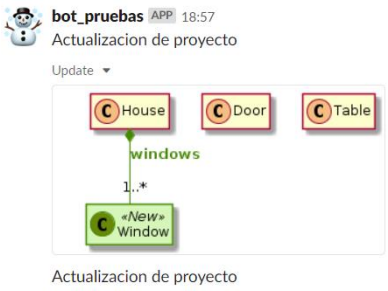


Ilustración 13 – Imagen al añadir un elemento al proyecto

- Eliminar

Ilustración 14 – Imagen al eliminar un elemento

Estas comunicaciones se realizan gracias a la librería facilitada por Python, SlackBot.

5.3 Bot y el usuario

Cabe profundizar en este apartado ya que se ha establecido esta comunicación, pero no se ha explicado nada sobre ella. Esta comunicación se basa en sí, en el encapsulamiento de peticiones post, a una Api que establece Slack por defecto.

La librería permite poder realizar todo tipo de acciones, desde recibir información de los usuarios del canal, envió de imágenes, textos… todo ello encapsulado en peticiones que van dirigidas a la Api de Slack, y esta a su vez, lo hace llegar a al canal.

La interacción entre el usuario es muy sencilla, como ya se ha mencionado anteriormente se hará uso del @botModelado, seguido del comando, esto hará que el bot comience a realizar las preguntar para cumplimentar la información requerida. También existe la posibilidad para ciertas acciones, que seguido del comando se escriba la información necesaria, hará que el bot directamente lance la petición en caso de cumplimentar toda la información necesaria.

El bot hace uso de varias librerías en Phyton las cuales hay que destacar:

- SlackClient: es la encargada de facilitar los recursos al lenguaje para poder comunicarse con Slack.

- SlackBot: nos aporta las funciones necesarias para poder crear un bot y sincronizarlo con Slack.

- Myqsl: para poder tener una gestión de los usuarios el bot almacena ciertos datos de los usuarios y de la aplicación con esta librería.

Gracias a la gestión de hilos facilitada por la librería Threading, podemos gestionar tanto los comandos del usuario, como las actualizaciones de los proyectos y las votaciones en cualquier momento.

5.4 VPS

El elemento donde se albergan el bot y el webservice para que el funcionamiento del sistema global sea a tiempo real y totalmente accesible. En el interior del servidor podremos ver que ocurre y como nos llegan las contestaciones de los usuarios. Además, veremos cómo se ha podido acceder, modificar al código los elementos y destacar la conexión interna que se produce entre el bot y la base de datos

También se ha tenido que preparar el entorno donde se alojan el webservice y el bot:

- Instalar el paquete LAMP: este paquete compuesto a su vez por los componentes necesarios para la instalación y funcionamiento de Mysql, Php, y Apache.

- PhpMyAdmin: será el encargado de dotar la interfaz gráfica a Mysql pudiendo gestionar toda la creación y gestión de la base de datos.

- Python 2.7: necesario para que tanto el webservice cómo el bot funcione correctamente ya que el lenguaje del código de estos componentes es ese.

- Virtualvenv: para encapsular cada entorno de desarrollo de los componentes del sistema y para que ninguna dependencia requerida afecte negativamente en alguno de los elementos que se alojan en el servidor, se crea un entorno virtual de Python para que se instalen independientemente las dependencias necesarias, de esta manera el sistema contendrá las dependencias generales pero cada entorno tendrá las suyas aislando si cada proyecto.

- Screen: este paquete nos ayuda a gestionar las sesiones en el propio servidor pudiendo crear varias sesiones de ellas pudiendo lanzar los elementos y gestionarlos en diferentes “pantallas”.

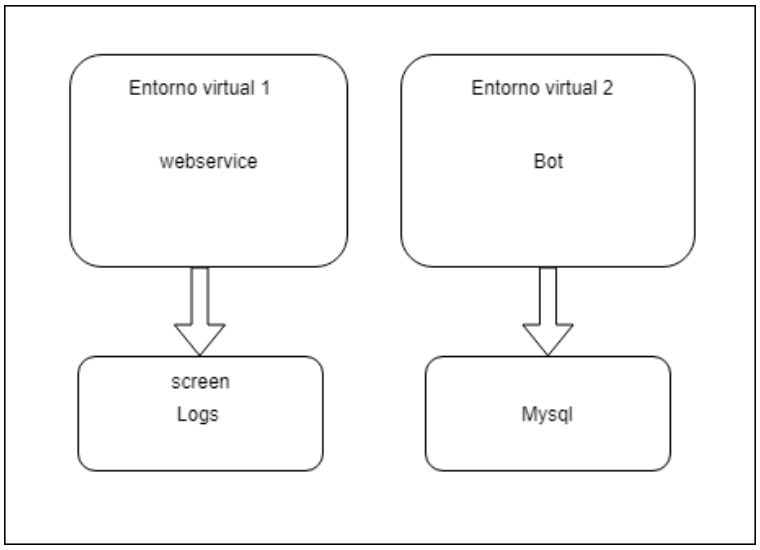


Ilustración 15 - Integración en VPS

Como se puede ver en la imagen 13, así es como quedarían los componentes alojados en el VPS. Dos entornos virtuales diferentes, cada uno con el webservice y el bot. Además, se aloja una base de datos mysql que conecta con el bot. Y paralelamente, en otro screen, el log con las peticiones que recibe el webservice.

5.4.1 Webservice

Gracias al servicio web podremos enviarle la respuesta que el usuario ha elegido. Para la implementación del Webservice se ha necesitado la librería Flask. Este framework de Python se usa para el desarrollo de aplicaciones web bajo el patrón MVC.

Este framework es ideal para la implementación del webservice ya que no va a tener mucha complejidad, el simple uso de recibir las peticiones, sacar el mensaje y reenviarlo al bot.

Necesitaremos crear un entorno virtual y un screen para poder visualizar toda la información relativa, el webservice se puede lanzar y visualizar en ese screen toda la información.

Este tipo de webservices se llaman Rest, esta de tecnología se basa en la creación de rutas para realizar tareas diferentes. Un ejemplo, si queremos que cuando un mensaje llegue y se envíe al bot, para esa acción Flask asignara una ruta donde enviar la petición correspondiente, si por el contrario lo que queremos es otra acción se enviara la petición a otra ruta diferente.

Gracias a la estructura interna del webservice, este sabe en todo momento que acción realizar para cada tipo de petición que nos llegue. En nuestro caso el funcionamiento será muy simple, solo hay una ruta donde hacer las peticiones, de tipo POST, y la acción se basará en rescatar el valor de la respuesta del usuario recibido en JSON, formar el mensaje y gracias a la librería SlackClient podremos enviarlo al bot el cual estará esperando la

respuesta. De esta manera podemos simular la contestación del usuario y el bot proseguir con la funcionalidad que esté haciendo.

Como dato a añadir, se implementará además un controlador de errores para que en caso de recibir una petición errónea haya una contestación negativa por parte del webservice.

Este webservice está desarrollado en lenguaje Python e implementa una Liberia llamada Flask que hace posible la creación de un webservice de tipo Rest donde a través de la URL recibimos las respuestas de los usuarios, este se las transmite de nuevo a Slack donde estará el Bot escuchando continuamente alguna posible respuesta de algún usuario que este usando la aplicación de modelado en Slack.

Sabido el tipo de webservice que es, Rest, se accede por el por medio de peticiones. Flask el framework que usa Python para diseñar este tipo de aplicaciones, abre un puerto en el servidor por donde recibirá las peticiones.

En este caso, hemos establecido el puerto por defecto que ofrece el framework, el 5000. Gracias al comando netstat -l podemos visualizar si está el puerto escuchando las peticiones:

```
root@vps620848:~# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:5000                  *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
```

Conociendo el puerto de escucha y la URL se puede establecer la conexión con el webservice. Cuando se ha implementado el webservice este devuelve la url por la cual se puede establecer la conexión:

http://51.77.145.141:5000/slack/message_options

Se puede ver que el comienzo es el protocolo por el cual se accede, la segunda parte pertenece a la IP del servidor VPS, seguido del puerto y por último la ruta relativa que levanta Slack para poder gestionar las diferentes acciones, ya que según el número de rutas se podrán realizar tantas acciones.

Además, gracias la herramienta, que permite abrir tantas terminales como queramos, screens, podremos dejar visible el modo debug de Slack el cual nos informa de las diferentes peticiones que llegan:

```
185.156.177.20 - - [01/May/2019 06:02:33] code 400, message Bad HTTP/0.9 request
185.156.177.20 - - [01/May/2019 06:02:33] "/*\x80Cookie: mstshash=Administr" 400 -
196.52.43.87 - - [01/May/2019 08:28:35] "GET / HTTP/1.0" 404 -
92.246.76.138 - - [01/May/2019 15:24:28] code 400, message Bad HTTP/0.9 request
92.246.76.138 - - [01/May/2019 15:24:28] "/*\x80Cookie: mstshash=Administr" 400 -
198.108.67.48 - - [02/May/2019 11:33:55] "GET / HTTP/1.1" 404 -
173.72.101.149 - - [02/May/2019 14:28:21] code 400, message Bad HTTP/0.9 request
173.72.101.149 - - [02/May/2019 14:28:21] "$(reboot -f)" 400 -
185.156.177.20 - - [02/May/2019 15:42:18] code 400, message Bad HTTP/0.9 request
185.156.177.20 - - [02/May/2019 15:42:18] "/*\x80Cookie: mstshash=Administr" 400 -
196.52.43.112 - - [02/May/2019 18:36:19] "GET / HTTP/1.0" 404 -
106.75.84.197 - - [03/May/2019 00:58:44] "GET / HTTP/1.0" 404 -
112.64.199.58 - - [03/May/2019 16:02:50] "GET / HTTP/1.0" 404 -
185.156.177.156 - - [03/May/2019 19:10:01] code 400, message Bad HTTP/0.9 request
185.156.177.156 - - [03/May/2019 19:10:01] "/*\x80Cookie: mstshash=Administr" 400 -
62.4.14.206 - - [04/May/2019 15:18:38] "GET / HTTP/1.1" 404 -
```

Ilustración 16 – Debug de Flask

Como se puede apreciar en la imagen se pueden ver claramente 4 elementos en las peticiones:

- Ip origen: de donde se lanza la peticion.
- Fecha y hora de cuando se ha recibido
- Protocolo y tipo de peticion
- StatusCode: el resultado que ha obtenido la peticion que ha llegado

A su vez, este webservice debe de devolver la informacion recibida al bot. Aunque los dos componentes se encuentran alajodos en el mismo sitio fisico, el webservice se tiene que comuncicar con el Bot a traves de la API de slack. Añadiendo en el codigo fuente del webservice la autentificacion del bot de Slack, podemos enviar a la API la respuesta y esta a su vez se la hace llegar al bot.

Hay una conexión que si se realiza internamente, es la comunicación que hay entre el bot y Mysql. La liberia para python mysql permite atraves de la funcion connect() establecer una conexión a un base de datos remota, en este caso, la base de datos al alojarse fisicamente en el mismo sitio, podremos configurar esa conexión para que directamente sea local (localhost)

```
MySQLConnection(host="localhost", user="***",password= "***", database="bot")
```

Esta conexión se establece cuando se recibe un mensaje de un usario, para tener un control del mismo y sobre todo integrar los ids del usuario en Slack y SOCIO. Una vez que el usuario recibe la informacion o realiza las acciones pertinentes esta conexión se cierra, ya que al no cerrar la conexión podria dar fallos de socket al volver a conectarse.

5.4.2 Desarrollador y VPS

La forma de conectarnos remotamente al servidor es vía SSH. Gracias a este protocolo y abriendo el puerto 22 del puerto podremos conectarnos al servidor. Además, gracias a diferentes paquetes podremos editar, subir o modificar el código fuente de los proyectos desde el mismo.

Con git, podremos bajarnos los códigos fuentes desde el repositorio de GitHub, y además podremos sincronizarlo en caso de haber modificado parte del código. Para la modificación del código podemos encontrar varios paquetes, en este caso se ha usado vim, un editor de texto a nivel de consola con el que podremos sin problemas editar el código del proyecto.

6 Integración, pruebas y resultados

Como hemos visto en el diseño y desarrollo, hemos tenido que tratar cada sistema de manera individual, explicándolo y desarrollándolo para más tarde integrar y gestionar las diferentes comunicaciones.

Para la integración y pruebas del sistema se han realizado de la misma manera, se testearán de una manera individual los diferentes componentes, dándole más importancia a la correcta comunicación entre los subsistemas para que el traspaso de información sea la correcta.

Además, se han realizado tratamientos de errores para que en caso de haber algún error en algunas de las comunicaciones del sistema este no se vea afectado, sea notificado ese error al usuario y el sistema pueda seguir realizando su función.

6.1 ChatBot con Socio

La integración de los sistemas es relativamente sencilla. El chatbot alojado en el vps, e internamente aislado por un entorno virtual será capar de enviar y recibir peticiones vía http, por medio de peticiones post y put.

Estas peticiones ya predefinidas en el webservice de SOCIO se mandarán al api alojada en uno de los servidores de la UAM, esta, dependiendo del tipo y formato que se ha enviado devolverá al chatbot la información junto con un mensaje de que todo ha salido bien, y el error en caso contrario.


```
<Response [200]> {"projectList":[{"isBranch":false,
 "closeBranchs":["Branch group: Groupbranchvota\n\t+ Selected:
 slack/Pablo/branchvota\n\t All branchs:"],"isOpen":true,
 "visibility":"PUBLIC","name":"TestVotacion","admin":{"nick":"Pablo",
 "name":"Pablo","channel":"slack","id":1},"id":149,"type":"metamodel",
 "usersCanRead":[],"openBranchs":[{"branchGroup":"Groupbranchvota2",
 "name":"slack/Pablo/branchvota","id":150}],"createDate":1555663868112,
 "usersCanEdit":[]}]}
<Response [500]> Timer already cancelled.
```


Ilustración 17 – Ejemplo de devolución de SOCIO

En la imagen 15, podemos ver entre los caracteres [] el estado de la petición que en el primer caso es correcta, y en el segundo fallida. Entre {} el JSON con el contenido de la información, en este caso con la lista de proyectos que tiene un usuario, este mensaje luego será tratado por el bot para hacérselo llegar al usuario.

Para una buena gestión en integración entre ambos sistemas por cada petición se han visto los diferentes códigos de estado que generan las comunicaciones y un trato para los mismos, para cuando haya casos erróneos el sistema del chatbot pueda gestionarlos sin ningún inconveniente.

6.2 Chatbot con Slack

La comunicación entre ambos es muy parecida a la establecida entre el Bot y SOCIO ya que en él participan las mismas tecnologías, pero en diferentes entornos. Slack gestiona también las comunicaciones entre el bot y usuarios gracias a su api.

El bot lanza a través de código encapsulado un tipo de petición que desencadenara una acción en el motor de Slack y obtendrán los resultados. Al estar el código, encapsulado la gestión de errores es resuelta por la librería. Así mismo, la única gestión que se ha tenido que realizar es la devolución de un mensaje al usuario informando de la producción de un error interno.

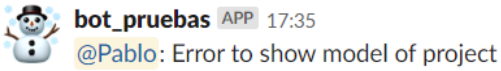


Ilustración 18 – Ejemplo información de error

Al ser esta, junto con la de socio, una de las conexiones principales, se han establecido unos gestores para que, en casos de error, el bot no se vuelva inaccesible. En muchas ocasiones con algunos softwares orientados a comunicaciones cuando estos sufren algún fallo en la comunicación, se caen, volviéndose inaccesibles hasta que se resuelva el error y vuelvan a lanzarse.

Para el chatbot de Slack, se ha integrado en el VPS un script que analiza si el bot está corriendo o no. Este script se encargará de ver si hay algún proceso con el PID que se obtuvo al lanzarse el bot.

Gracias al Crontab del sistema podemos definir que cada minuto se esté lanzado este script para volver a lanzar el main.py que se encuentra el bot.

6.3 Chatbot con Webhooks

Al tratarse como en las conexiones anteriores de la misma tecnología, estos dos sistemas se integran y se resuelven las conexiones con los mismos de la misma manera introduciendo una peculiaridad, entran en juego un webservice.

Webhooks, una Api que, como SOCIO, está a la espera de la llegada de una petición, por medio de la URL que nos facilita. Si la petición que nos llega es una petición para el envió de un mensaje interactivo informativos, se envía sin problemas para la api de Slack automáticamente al estar integrada en la misma, pero cuando es una petición con interacción, se ha tenido que integrar un webservice alojado en el VPS para recoger esas contestaciones y derivarlas al bot.

El bot sabe qué tipo de mensaje envía, al enviar un mensaje con interacción para un usuario aísla la conversación en un hilo con el fin de no perder su respuesta y no confundir en caso de llegar varias respuestas de diferentes usuarios a la vez

6.4 Chatbot con Mysql

Esta integración es muy sencilla, gracias a la librería mysql, el chatbot establece una conexión local con la base de datos previamente creada ya que se alojan en el mismo sitio.

Siempre que se abre una conexión y se ha recogido o insertado la información se cierra la conexión para no generar conflictos de accesos.

6.5 Pruebas

Al igual que para la integración, el testeo del chatbot se realizaron individualmente las pruebas para cada elemento del sistema y las comunicaciones con las que trabaja. Al ser un proyecto el cual tiene partes de desarrollo de código como de comunicación en los sistemas podemos diferenciar dos tipos de pruebas que se han realizado, comunicaciones y desarrollo.

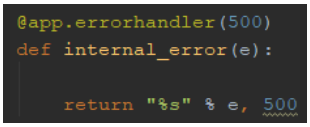
En sí, no solo se ha tenido que programar el chatbot si no que el webservice también entra en el tipo de pruebas para el desarrollo. Para realizar estas pruebas, no hemos puesto un perfil de ver el correcto funcionamiento del sistema, si no ver en que situaciones el sistema se puede ver afectado o comprometido.

Ahora se ira mencionando los diferentes elementos que componen el chatbot:

- Webservice: para comprobar el correcto funcionamiento se ha probado a enviar al webservice cierto tipo de peticiones, las cuales tenían alguna peculiaridad:

    - Peticiones al webservice con una URL inexistente: en estos casos, al enviar una petición a una URL inexistente el propio medio por el cual se envías las peticiones automáticamente debe de responder la inaccesibilidad del lugar.

    - Peticiones al webservice con peticiones vacías, información incorrecta, o codificación de los datos intangible: para estas peticiones se ha generado un controlador proporcionado por la librería FLASK.



    - Este controlador se indica en cada función que se desarrolle, el cual tiene la característica de poder configurar un mensaje personalizado del error. Además, hace que el sistema no caiga por este tipo de peticiones, haciendo que el usuario reciba el mensaje de error y que el sistema pueda continuar recibiendo peticiones.

- Webhooks y Slack: estos dos sistemas, aunque forman parte del chatbot al ser material de terceros se encargan ellos mismo de la gestión de errores. De la misma

manera que gestionan los errores y se ahorra en desarrollos de tratamiento de los mismos, no hay manera de saber cuándo uno de estos sistemas está caído, ya que solo devolverán error en caso de envíos de información erróneas. Al igual que con el webservice se han enviado peticiones de diferentes maneras, con información errónea, con información intangible, falta de información y siempre estos sistemas comunican el tipo de error y donde reside el mismo.

- Mysql: gracias a las pruebas, se pudo encontrar un error que el sistema de gestión de los datos. Se incluye el control de información por la necesidad de convertir y englobar los ID's entre SOCIO y Slack, pero siempre para usuarios que pertenecían a Slack, no a otras plataformas.

  Como socio es multiplataforma, en el momento que un usuario de otra plataforma nos invitaba a participar en un proyecto, al no tener constancia de ese usuario el sistema se caía, la solución es simple, por cada petición que nos llega de cierto tipo, se busca el usuario en la base de datos y en caso de no existir se crea uno nuevo, esto hace que siempre tengamos constancia de proyectos y usuario que afectan o directa o indirectamente al sistema.

- Chatbot Slack: al ser el pilar del proyecto, para este elemento se han realizado diferentes pruebas tanto de desarrollo como de comunicaciones. A nivel de desarrollo podemos destacar varios casos:

  - Comandos: una de las partes críticas del chatbot es la conversación con los usuarios, este siempre debe entender lo que el usuario le ha transmitido. Aunque la comunicación entre el usuario es super simple ya que solo tienen que llamar al bot junto con comando y en algunos casos con las opciones, el usuario puede introducir erróneamente los datos.

    Al realizar ese tipo de pruebas introduciendo un comando inexistente el sistema se caía. Se tuvo que crear un controlador para informar al usuario de las acciones que puede realizar y lo más importante, el sistema no se cae en este tipo de situaciones.

    Cuando el usuario introduce un comando correctamente, pero con opciones diferentes, pasa exactamente igual que si manera un comando inexistente, se le informa de que manera se realiza esa acción.

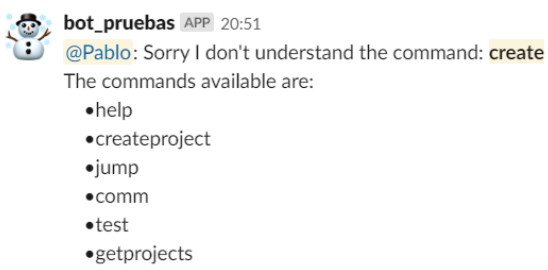


Ilustración 19 – Gestor de errores

- Gestión de hilos: una de las bases fundamentales de un chatbot, es la gestión de múltiples conversaciones. Para asegurar que esto no fuera un problema, se crearon varios usuarios diferentes, y se probaron en diferentes equipos y plataformas ya que a Slack se puede acceder vía app y http, al igual que en diferentes sistemas como puede ser Android y w10. Los resultados han sido satisfactorios gracias a la gestión de conversaciones por medios de hilos.

- Interacciones de los comandos y errores internos: una vez que unitariamente se integraron el traspaso de información correctamente, era necesario implantar el comando en el sistema, y se probaban las diferentes situaciones que podían darse.

  Unos ejemplos:

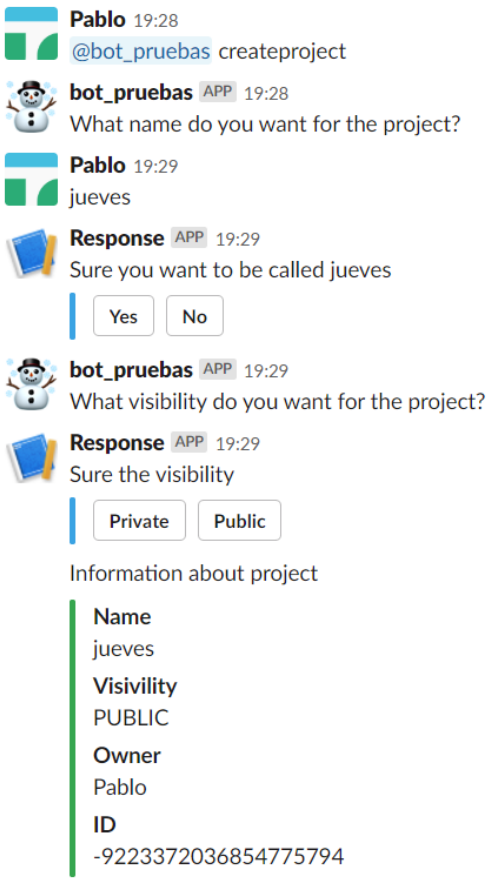


Ilustración 20 – Proceso de crear un proyecto

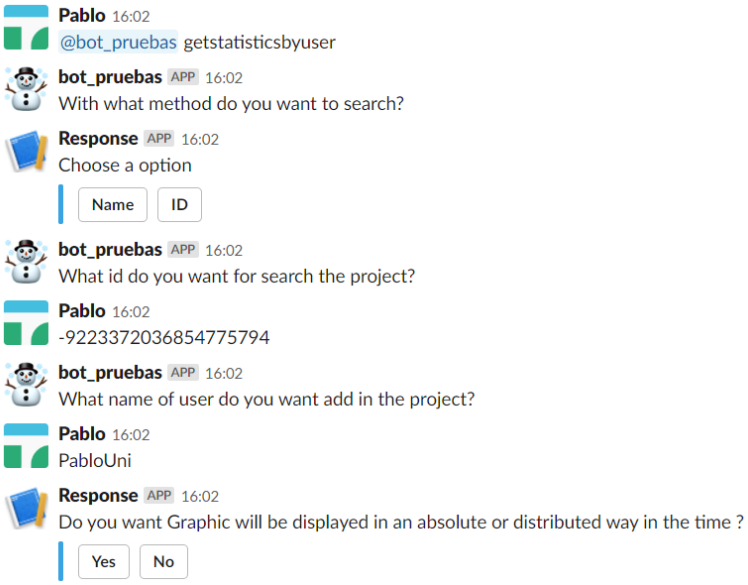


Ilustración 21 – Proceso de obtener estadísticas

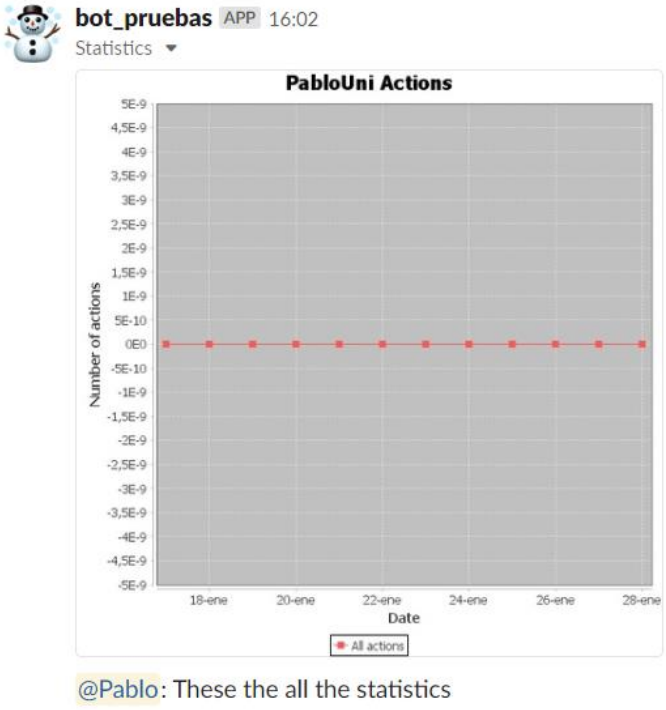


Ilustración 22 – Imagen con las estadísticas del usuario

A nivel de comunicaciones podemos encontrarnos tanto con el lanzamiento como con recepción de las peticiones. Antes de implementar una petición a algún sistema, esta ha sido lanzada unitariamente. Se lanzaba no solo la petición para obtener un resultado satisfactorio, sino que lanzaban peticiones con ciertos errores para ver cómo actuar en esas situaciones y saber transmitir al usuario o al sistema ese fallo.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

En este trabajo se ha desarrollado un chatbot para el trabajo colaborativo en Slack donde he podido comprobar la capacidad y utilidades que puede tener un chatbot, y el potencial que puede llegar a tener SOCIO en la red Slack. Es una aplicación que encaja a la perfección ya que esta red es usada como chat de carácter empresarial debido a las muchas ventajas e integraciones que tiene.

El mundo de las aplicaciones en la red es un plus que le suma al usuario al no tener que estar gastando recursos de su ordenador, este plus más los añadidos anteriormente hacen ver como avanza el mundo de las tecnologías en ámbitos como las comunicaciones o la automatización de tareas.

Entrando mas profundamente en lo que ha sido internamente el proyecto, que se ha desarrollado un sistema de comunicación entre una aplicación de modelado colaborativo con características no encontradas en otras aplicaciones similares, como puede ser la gestión a través de las redes sociales, y más hoy con el uso que se le da a las mismas.

Otro tema del cuál se ha profundizado has sido el tema de las comunicaciones entre sistemas, hoy en día fundamentales. Gracias a ciertas tecnologías es posible el cambio de información, pudiendo integrar en sí, sistemas más complejos. Fomentando la usabilidad y facilitando al usuario la experiencia de uso más sencilla y productiva.

7.2 Trabajo futuro

Hoy el día el mundo de los chatbots es una tecnología muy usada y de la cual ya hay desarrollos realizados. A estos, se les puede llegar aplicar inteligencia artificial para diferentes usos, uno de los posibles trabajos futuros seria realizar una IA capaz de registrar las acciones comunes por el usuario, para facilitarle las acciones.

Otro desarrollo y mejora sería el análisis sintáctico, pudiendo desalojar los patrones para realizar las acciones de gestión de la aplicación y hacer más amena y cercana la interacción con el usuario.

La posibilidad de mandar mensajes de bot en Slack hace posible la capacidad de poder implementar interacciones entre el chatbot y el usuario por voz, ayudando a la accesibilidad de la aplicación de modelado colaborativo.

Referencias

[1] LAS REDES SOCIALES: UNA NUEVA HERRAMIENTA DE DIFUSIÓN, Harold Hütt Herrera, ISSN: 1021-1209 / 2012 [Última fecha acceso: 11/06/2019]

[2] MODELADO COLABORATIVO EN LENGUAJE NATURAL A TRAVÉS DE REDES SOCIALES, Sara Perez Soler 2018 [Última fecha acceso: 11/06/2019]

[3] The Rise of Social Bots, EMILIO FERRARA, ONUR VAROL, CLAYTON DAVIS, FILIPPO MENCZER, AND ALESSANDRO FLAMMINI, Communications of the ACM Volume 59 Issue 7 [Última fecha acceso: 11/06/2019]

[4] https://www.puromarketing.com/12/28641/variados-complejos-usos-bots-podrian-tener-para-marcas-empresas.html [Última fecha acceso: 11/06/2019]

[5] https://trends.google.es/trends/explore?date=today%205-y&geo=ES&q=chatbot [Última fecha acceso: 11/06/2019]

[6] http://mktefa.ditrendia.es/blog/los-mejores-chatbots-de-2018-son-de-banca-y-salud [Última fecha acceso: 11/06/2019]

[7] COMPORTAMIENTO ADAPTABLE DE CHATBOTS DEPENDIENTE DEL CONTEXTO, Adaptable de Chatbots Dependiente del Contexto, Revista Latinoamericana de Ingeniería de Software: 115-136, ISSN 2314-2642, Rodríguez, J., Merlino, H., Fernández, E. 2014. [Última fecha acceso: 11/06/2019]

[8] https://tfg-uam-eps.slack.com/apps/A0F81FKT8-codeship [Última fecha acceso: 11/06/2019]

[9] https://obie.ai/ [Última fecha acceso: 11/06/2019]

[10] https://tfg-uam-eps.slack.com/apps/A2RPP3NFR-jira-cloud [Última fecha acceso: 11/06/2019]

[11] https://cai.tools.sap/ [Última fecha acceso: 11/06/2019]

[12] https://tfg-uam-eps.slack.com/apps/A08MGBLJV-zeplin [Última fecha acceso: 11/06/2019]

[13] https://tfg-uam-eps.slack.com/apps/A8GBNUWU8-github [Última fecha acceso: 11/06/2019]

[14] https://www.genmymodel.com/ [Última fecha acceso: 11/06/2019]

[15] https://www.gliffy.com/ [Última fecha acceso: 11/06/2019]

[16] http://diagramo.com/ [Última fecha acceso: 11/06/2019]

[17] https://www.google.es/intl/es/docs/about/ [Última fecha acceso: 11/06/2019]

[18] https://products.office.com/es-es/home [Última fecha acceso: 11/06/2019]

[19] https://www.dropbox.com [Última fecha acceso: 11/06/2019]

[20] https://wetransfer.com/ [Última fecha acceso: 11/06/2019]

[21] Generating Typed Dependency Parses from Phrase Structure Parses . M. Marneffe, B. Maccartney y C. Manning. En: Proc. LREC, https://nlp.stanford.edu/software/lex-parser.shtml [Última fecha acceso: 11/06/2019]

[22] Uribe Saavedra, F., Rialp Criado, J., & Llonch Andreu, J. (2013). El uso de las redes sociales digitales como herramienta de marketing en el desempeño empresarial. Cuadernos De Administración, 26(47), 205-232. Recuperado a partir de https://revistas.javeriana.edu.co/index.php/cuadernos_admon/article/view/7105 [Última fecha acceso: 11/06/2019]

[23] https://planetachatbot.com/evoluci%C3%B3n-de-los-chatbots-48ff7d670201 [Última fecha acceso: 11/06/2019]

Glosario

API Application Programming Interfaz